Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

*К защите допустить:*

И.О. Заведующего кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОЕКТИРОВАНИЕ БАЗА ДАННЫХ ДЛЯ ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ВИКТОРИН С НЕЧЕТКИМИ КРИТЕРИЯМИ ПРИНЯТИЯ ОТВЕТА**

БГУИР КП 1-40 04 01 033 ПЗ

Студент Г. К. Лямцев

Руководитель А. В. Давыдчик

Минск 2025

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc212848989)

[1 Обзор существующих аналогов 6](#_Toc212848990)

[1.1 JKLM.fun 6](#_Toc212848991)

[1.2 Jackbox Games 6](#_Toc212848992)

[1.3 Kahoot! и Quizizz 7](#_Toc212848993)

[1.4 Выводы 8](#_Toc212848994)

[2 Функциональные требования и инструменты разработки 9](#_Toc212848995)

[2.1 Функциональные требования к системе 9](#_Toc212848996)

[2.2 Выбор СУБД 10](#_Toc212848997)

[3 Проектирование базы данных 14](#_Toc212848998)

[3.1 Инфологическая модель 14](#_Toc212848999)

[3.2 Даталогическая модель 16](#_Toc212849000)

[4 Разработка базы данных 21](#_Toc212849001)

[4.1 Физическая модель базы данных 21](#_Toc212849002)

[4.2 Создание таблиц 21](#_Toc212849003)

[4.3 Запросы 23](#_Toc212849004)

[Заключение 25](#_Toc212849005)

[Список литературных источников 26](#_Toc212849006)

[Приложение А (обязательное) Справка о проверке на заимствование 27](#_Toc212849007)

[Приложение Б (обязательное) Листинг программного кода 28](#_Toc212849008)

[Приложение В (обязательное) Функциональная схема алгоритма, реализующего программное средство 29](#_Toc212849009)

[Приложение Г (обязательное) Блок схема алгоритма, реализующего программное средство 30](#_Toc212849010)

[Приложение Д (обязательное) Графический интерфейс пользователя 31](#_Toc212849011)

[Приложение Е (обязательное) Ведомость документов 32](#_Toc212849012)

# Введение

В современном цифровом мире интерактивные формы досуга и обучения, такие как онлайн-викторины, приобретают всё большую популярность. Они используются для образования, корпоративных мероприятий, стриминга и просто как увлекательный способ провести время с друзьями. Однако большинство существующих платформ обладают жёсткой системой проверки ответов, где опечатка, синоним или иной порядок слов могут привести к незасчитанному, по сути, правильному ответу, что снижает удовольствие от игры и ограничивает креативность создателей вопросов.

Целью данного курсового проекта является проектирование и разработка базы данных для веб-приложения «QuizFuzz» – платформы для проведения многопользовательских викторин в реальном времени с поддержкой нечётких критериев принятия ответов. Ключевой особенностью системы является реализация гибкого механизма оценки ответов пользователей, который учитывает возможные опечатки, синонимы, фонетическое сходство и даже семантическую близость, что делает игровой процесс более комфортным и интуитивно понятным.

Основные задачи разработки включают:

1 Анализ предметной области и существующих аналогов для выявления их достоинств и недостатков.

2 Формирование функциональных требований к системе и определение подходящих инструментов для разработки.

3 Проектирование базы данных, включая разработку инфологической и даталогической моделей, адекватно отражающих сложную бизнес-логику приложения.

4 Создание физической модели базы данных с использованием выбранной системы управления базами данных (СУБД).

5 Описание ключевых запросов, обеспечивающих функционирование приложения.

Проектирование базы данных является критически важной частью разработки, так как она должна обеспечивать не только надежное хранение структурированных данных о пользователях, вопросах, игровых сессиях и ответах, но и высокую производительность при выполнении сложных запросов в реальном времени, а также поддерживать механизмы нечёткого поиска и аналитики. Грамотно спроектированная база данных станет основой для создания масштабируемого, производительного и отказоустойчивого приложения, способного выдерживать нагрузку от тысяч одновременных пользователей.

1. ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ

Прежде чем приступить к проектированию собственной системы, необходимо провести анализ существующих на рынке решений, реализующих схожую функциональность. Это позволяет выявить их сильные и слабые стороны, определить общепринятые практики и выделить ниши для внедрения инноваций. Проведенный обзор поможет сформулировать конкурентные преимущества разрабатываемого приложения QuizFuzz и избежать ошибок, допущенных в существующих продуктах. В качестве аналогов были рассмотрены такие платформы, как JKLM.fun, Jackbox Games, Kahoot! и Quizizz.

* 1. JKLM.fun

JKLM.fun – это минималистичная веб-платформа для многопользовательских игр-викторин в реальном времени, которая стала одним из ключевых ориентиров для QuizFuzz.

Функциональность:

1 Создание приватных комнат по короткой ссылке.

2 Поочерёдные раунды с вопросами, на которые игроки вводят текстовые ответы.

3 Базовый механизм «нечёткого» сопоставления ответов, допускающий небольшие опечатки.

Преимущества:

1 Простота и минимализм пользовательского интерфейса, позволяющие быстро начать игру.

2 Низкий порог входа – не требуется регистрация.

3 Эффективная система приглашений в комнаты.

Недостатки:

1 Крайне ограниченный и неконфигурируемый механизм проверки ответов, не поддерживающий синонимы, фонетику или семантику.

2 Отсутствие системы пользователей, статистики, лидербордов и модерации контента.

3 Невозможность создавать собственные вопросы внутри платформы.

Таким образом, JKLM.fun представляет собой удобный, но функционально ограниченный инструмент, решающий базовую задачу проведения викторины без глубокой проработки гибкости проверки и аналитики.

* 1. Jackbox Games

Jackbox Games – это серия популярных party-игр, включающих викторины, которые транслируются на большом экране, а игроки участвуют со своих смартфонов.

Функциональность:

1 Набор разнообразных игровых пакетов с уникальной механикой.

2 Встроенные системы создания пользовательского контента (в некоторых играх).

3 Высококачественный визуальный ряд и озвучка.

Преимущества:

1 Полировка и высокое качество контента.

2 Интуитивный и весёлый геймплей, ориентированный на вечеринки.

3 Мощные встроенные инструменты модерации и цензуры.

Недостатки:

1 Является коммерческим продуктом с необходимостью покупки игровых пакетов.

2 Закрытая экосистема без возможности самостоятельного создания и модерации вопросов сообществом в единой базе.

3 Отсутствие гибкой системы проверки ответов, аналогичной нечёткому поиску QuizFuzz.

Платформа Jackbox служит отличным примером организации социального геймплея и работы комнат, но её модель не подходит для создания открытой платформы с пользовательским контентом.

* 1. Kahoot! и Quizizz

Kahoot! и Quizizz — это образовательные платформы для проведения викторин, широко используемые в учебных заведениях и корпоративной среде.

Функциональность:

1 Создание викторин с различными типами вопросов (выбор ответа, верно/неверно).

2 Публичные и приватные игровые сессии.

3 Подробная статистика и отчеты по результатам игроков.

4 Обширные библиотеки готового контента.

Преимущества:

1 Мощные аналитические и статистические инструменты.

2 Фокус на образовательном процессе и удобство для преподавателей.

3 Высокая стабильность и масштабируемость.

Недостатки:

1 Проверка ответов основана исключительно на выборе заранее заготовленных вариантов, что полностью исключает текстовый ввод и, как следствие, возможность реализации нечёткого сопоставления.

2 Отсутствие механик, характерных для развлекательных викторин (подсказки, динамическое начисление очков за скорость).

Данные платформы идеально решают задачи образования, но их игровая механика слишком ограничена для создания гибкой развлекательной системы с открытыми ответами.

* 1. Выводы

Проведенный анализ показал, что существующие решения либо предлагают простоту и минимализм в ущерб функциональности (JKLM.fun), либо являются закрытыми коммерческими продуктами с фиксированным контентом (Jackbox), либо ориентированы исключительно на формат тестов с множественным выбором (Kahoot!, Quizizz). Ни одна из рассмотренных платформ не сочетает в себе возможность пользовательского генерирования контента, мощную систему модерации, глубокую аналитику и, что не менее важно, гибкий механизм нечёткой проверки текстовых ответов. Это подтверждает актуальность и новизну разрабатываемого веб-приложения QuizFuzz, призванного заполнить выявленную рыночную нишу.

1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ И ИНСТРУМЕНТЫ РАЗРАБОТКИ
   1. Функциональные требования к системе

Разрабатываемая система «QuizFuzz» представляет собой комплексное веб-приложение, предназначенное для организации и проведения многопользовательских викторин в реальном времени. Его ключевой особенностью является поддержка нечётких критериев оценки текстовых ответов, что значительно расширяет стандартные возможности подобных платформ.

Основные функциональные требования к системе включают следующие модули:

1 Модуль аутентификации и авторизации обеспечивает безопасный доступ пользователей к системе. Он должен предоставлять функционал регистрации по электронной почте и паролю, входа в систему и безопасного выхода, а также восстановления доступа к учетной записи. В рамках данного модуля реализуется ролевая модель управления доступом, включающая роли Гостя, Пользователя, Модератора и Администратора. Для защиты API необходимо использовать JWT-токены, а также предусмотреть механизмы защиты сессий и ограничения частоты запросов на аутентификацию.

2 Модуль управления игровыми комнатами отвечает за организацию виртуального пространства для проведения игр. Он позволяет пользователям создавать, редактировать и закрывать игровые комнаты. При создании комнаты должны настраиваться её ключевые параметры: тип доступа (публичный или приватный), лимит участников, условие завершения игры (по достижению определённого количества очков или числа раундов), а также система тегов для подбора вопросов. Для приватных комнат предусматривается генерация кодов доступа или уникальных ссылок-приглашений. Модуль должен предоставлять интерфейс для просмотра списка публичных комнат с возможностью их поиска и фильтрации.

3 Модуль игрового процесса является ядром системы и управляет проведением викторины. Его функционал включает организацию игровых сессий, состоящих из последовательных раундов. Система должна поддерживать различные типы вопросов: текстовые, с изображениями и аудио. Для каждого раунда устанавливается таймер, а также система динамических подсказок, которые открываются в заданные моменты времени. Игроки имеют возможность отправлять ответы неограниченное количество раз до истечения времени раунда, при этом модуль должен осуществлять защиту от флуда. Все события в реальном времени транслируются участникам с использованием технологии SignalR.

4 Модуль нечёткой проверки ответов реализует инновационный подход к оценке ответов игроков. Он включает конвейер предварительной обработки введённого текста, включающий нормализацию, приведение к нижнему регистру, удаление пунктуации и транслитерацию. Модуль поддерживает несколько стратегий проверки: точное совпадение без учёта регистра, вычисление редакционного расстояния по алгоритмам Левенштейна и Дамерау, проверку по заранее заготовленным алиасам и синонимам, фонетическое сравнение с использованием алгоритмов Double Metaphone и их аналогов для кириллицы, а также семантический анализ на основе векторных представлений слов и фраз. Реализуется механизм агрегации результатов от разных стратегий с назначением весов и порогов уверенности, а также логирование процесса проверки для последующего анализа.

5 Модуль управления контентом и модерации регулирует процесс наполнения системы вопросами. Пользователи получают возможность подавать собственные вопросы на модерацию через специальную форму, указывая тип вопроса, контент, правильный ответ, подсказки и теги. Для модераторов и администраторов предоставляется интерфейс очереди модерации, где вопросы проверяются, редактируются, утверждаются или отклоняются. Каждое действие модератора фиксируется в истории изменений. Модуль также включает управление системой тегов для категоризации контента.

6 Модуль статистики и аналитики собирает и представляет данные об активности пользователей. Он формирует персональную статистику для каждого игрока, включая количество сыгранных и выигранных игр, общую точность ответов, количество уникально угаданных вопросов, среднее время ответа и предпочтительные категории. На основе этих данных строятся глобальные, недельные и сезонные лидерборды. Пользователи имеют доступ к детальной истории своих игр с возможностью просмотра результатов по каждому раунду.

7 Административный модуль предоставляет инструменты для управления всей системой. В его функционал входит управление пользователями: назначение ролей, блокировка аккаунтов. Администраторы осуществляют управление системой тегов, настройку системных лимитов и параметров работы приложения. Модуль также обеспечивает просмотр аудит-логов, в которых фиксируются все значимые действия пользователей и администраторов в системе.

* 1. Выбор СУБД

Для успешной реализации проекта QuizFuzz, учитывая его требования к хранению сложноструктурированных данных, обеспечению целостности, поддержке транзакций и высокой производительности при одновременной работе тысяч пользователей, был выбран подход, основанный на использовании реляционной модели данных. Данный подход обеспечивает логическую целостность информации, удобство представления данных в виде таблиц и эффективность выполнения сложных запросов, необходимых для работы системы игровых сессий, статистики и нечёткого поиска.

Требования к СУБД для проекта QuizFuzz:

1 Поддержка сложных структур данных и связей. Данные системы включают множество взаимосвязанных сущностей (пользователи, вопросы, комнаты, сессии, ответы, оценки), которые требуют соблюдения целостности связей и сложных JOIN-запросов.

2 Гарантия целостности и согласованности данных (ACID). Операции, такие как подсчет очков, завершение раунда и обновление лидербордов, должны быть атомарными и согласованными, что исключает возможность возникновения конфликтов и потерь данных при высокой параллельной нагрузке.

3 Высокая производительность при обработке запросов. Система требует быстрого выполнения операций выборки и агрегации данных для формирования лидербордов, статистики пользователей и подбора вопросов для раундов.

4 Масштабируемость. Архитектура БД должна позволять горизонтальное и вертикальное масштабирование для поддержки растущего числа пользователей и игровых сессий.

5 Надёжность и отказоустойчивость. СУБД должна обеспечивать стабильную работу, механизмы репликации и резервного копирования для минимизации простоев.

6 Поддержка расширенных типов данных. Для реализации семантического поиска в будущем может потребоваться поддержка векторных типов данных (эмбеддингов).

7 Безопасность. Необходимы механизмы разграничения доступа, шифрования данных и обеспечения конфиденциальности пользовательской информации.

Анализ кандидатов:

Были рассмотрены следующие реляционные СУБД: PostgreSQL, MySQL и SQLite.

**2.2.1** PostgreSQL – это мощная, свободная объектно-реляционная система управления базами данных. Её разработка началась в 1986 году в Калифорнийском университете в Беркли, и с тех пор она прошла долгий путь, став одним из самых продвинутых и надежных решений в мире открытых СУБД. PostgreSQL строго следует стандартам SQL и поддерживает множество современных возможностей.

Преимущества:

1 Передовая функциональность. PostgreSQL предлагает богатый набор типов данных, включая нативные массивы, JSON/JSONB, XML, геопространственные данные и пользовательские типы. Особенно важно наличие расширения pgvector, которое позволяет эффективно хранить и выполнять поиск по векторным представлениям (эмбеддингам). Это критически важно для будущей реализации семантического модуля нечёткого поиска в QuizFuzz, где ответы можно будет сравнивать по смысловой близости.

2 Высокая производительность и оптимизация сложных запросов. PostgreSQL оснащен sophisticated оптимизатором запросов, который эффективно выполняет сложные JOIN-операции и агрегации, необходимые для формирования статистики пользователей и лидербордов. Поддержка расширенных методов индексирования (таких как GIN, GiST, BRIN) позволяет ускорить поиск по полнотекстовым данным, JSONB-полям и другим сложным структурам, что напрямую влияет на скорость подбора вопросов и анализа ответов.

3 Строгое соблюдение ACID и мощная система конкурентности. Для многопользовательской системы, такой как QuizFuzz, где десятки игроков могут одновременно отправлять ответы в одном раунде, механизм многоверсионного управления конкурентным доступом (MVCC) в PostgreSQL гарантирует, что транзакции по начислению очков и обновлению состояния игры будут изолированными и согласованными без блокировок на уровне таблицы.

4 Надёжность и отказоустойчивость. Поддержка синхронной и асинхронной репликации, механизмы point-in-time recovery (PITR) и активное сообщество делают PostgreSQL надежным выбором для проектов, где простои недопустимы.

Недостатки:

1 Требует более высокой квалификации администраторов. Для тонкой настройки производительности под конкретную нагрузку (настройка параметров shared\_buffers, work\_mem, эффективное использование партиционирования) требуются глубокие знания системы.

2 Потребление ресурсов. По умолчанию PostgreSQL может потреблять больше памяти и CPU, чем некоторые конкуренты, однако эта "прожорливость" является платой за её мощь и надежность, и для проекта уровня QuizFuzz эти затраты являются оправданными.

**2.2.2** MySQL – одна из самых популярных в мире открытых реляционных СУБД, изначально разработанная компанией MySQL AB и сейчас принадлежащая Oracle Corporation. Благодаря своей простоте, производительности и тесной интеграции со стеком LAMP (Linux, Apache, MySQL, PHP/Python/Perl), она стала стандартом де-факто для многих веб-приложений.

Преимущества:

1 Простота использования и настройки. MySQL известна своим быстрым временем развертывания и относительно низким порогом входа для администраторов. Огромное сообщество и обилие документации упрощают поиск решений типичных проблем.

2 Высокая производительность на операциях чтения. Для сценариев, преимущественно состоящих из операций SELECT (например, чтение вопросов, отображение списка комнат), MySQL с движком InnoDB демонстрирует очень хорошую скорость.

3 Репутация и распространённость. Широкая известность технологии снижает риски, связанные с её выбором, и упрощает поиск квалифицированных кадров для поддержки проекта.

Недостатки:

1 Ограниченная функциональность по сравнению с PostgreSQL. Исторически MySQL менее строго следовала стандартам SQL. Хотя ситуация улучшается, такие функции, как рекурсивные запросы (WITH RECURSIVE), развитые оконные функции и работа со сложными типами данных, в PostgreSQL часто реализованы более зрело и эффективно. Это могло бы ограничить возможности для сложной аналитики в QuizFuzz.

2 Ограничения в механизмах параллелизма. Хотя движок InnoDB значительно улучшил ситуацию, в сценариях с экстремально высокой конкурентной записью (когда сотни игроков одновременно отправляют ответы) MySQL в некоторых конфигурациях может показывать менее стабильную производительность по сравнению с PostgreSQL из-за различий в реализации MVCC.

**2.2.3** SQLite – это встраиваемая, автономная, бессерверная, транзакционная СУБД с нулевой конфигурацией. Её код находится в общественном достоянии, что делает её чрезвычайно популярной для встраивания в приложения, мобильные системы и браузеры.

Преимущества:

1 Простота развёртывания. SQLite не требует установки и администрирования отдельного серверного процесса. База данных представляет собой единственный файл на диске, что делает её идеальной для локальных приложений и прототипирования.

2 Нулевая нагрузка на администрирование. Отсутствие необходимости настраивать пользователей, права доступа, репликацию или резервное копирование на уровне СУБД (резервное копирование — это просто копирование файла).

Недостатки:

1 Отсутствие масштабируемости и сетевая архитектура. SQLite не поддерживает сетевые подключения и предназначена для использования одним приложением на одной машине. Эта архитектурная особенность делает её абсолютно непригодной для многопользовательского веб-приложения, такого как QuizFuzz, где тысячи клиентов должны одновременно обращаться к данным через веб-сервер.

2 Ограниченная производительность при высокой конкурентной записи. SQLite использует блокировку на уровне всей базы данных при операциях записи. Это означает, что если один игрок отправляет ответ и БД заблокирована на запись, все остальные игроки, пытающиеся отправить ответы в этот момент, будут вынуждены ждать. Это создает "бутылочное горлышко" и полностью исключает возможность реального времени для сотен одновременных пользователей.

**2.2.4** Обоснование выбора. Для высоконагруженного многопользовательского веб-приложения QuizFuzz, требующего сложных запросов, абсолютной целостности данных и возможности глубокого масштабирования, выбор пал на PostgreSQL.

Это решение обусловлено её беспрецедентной надёжностью, строгим соблюдением стандартов ACID, мощной поддержкой сложных типов данных и запросов, а также наличием критически важного расширения pgvector для будущей реализации семантического поиска. Её архитектура, основанная на многоверсионности (MVCC), идеально подходит для сценариев с высокой конкурентностью, характерных для игровых сессий. Хотя её настройка может быть сложнее, чем у MySQL, её производительность, функциональность и отказоустойчивость в условиях высокой нагрузки полностью соответствуют амбициозным требованиям проекта.

SQLite, в свою очередь, была сразу исключена из рассмотрения из-за фундаментальной архитектурной непригодности для многопользовательского клиент-серверного приложения.

Таким образом, PostgreSQL обеспечивает необходимый технологический фундамент для построения производительной, надежной и масштабируемой системы QuizFuzz, соответствующей всем заявленным в ТЗ требованиям как на текущий момент, так и для будущего развития.

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Проектирование базы данных является критически важным этапом создания системы QuizFuzz, поскольку от выбранной структуры хранения данных напрямую зависят производительность, надежность и масштабируемость всего приложения. В данном разделе представлены последовательные этапы проектирования БД: инфологическая модель, описывающая предметную область на концептуальном уровне, и даталогическая модель, определяющая структуру данных в терминах выбранной СУБД.

* 1. Инфологическая модель

Инфологическая модель представляет собой абстрактное описание предметной области, независимое от конкретной реализации в системе управления базами данных. Она фиксирует ключевые сущности, их атрибуты и взаимосвязи, отражая бизнес-логику приложения QuizFuzz.

Пользовательская подсистема:

1 User (Пользователь): Центральная сущность, содержащая учетные данные (email, хеш пароля) и основную информацию (имя, дата регистрации, последний вход).

2 Role (Роль): Справочник возможных ролей в системе: Admin, Moderator, User.

3 UserRole (Связь пользователя и роли): Обеспечивает назначение нескольких ролей одному пользователю.

Подсистема контента и модерации:

1 Question (Вопрос): Основная сущность, хранящая данные викторины: текст вопроса, тип (TEXT, IMAGE, AUDIO), правильный ответ, сложность, статус модерации.

2 QuestionType (Тип вопроса): Справочник допустимых типов контента.

3 Tag (Тег): Система категоризации для вопросов (например, "История", "Кино", "Наука").

4 QuestionTag (Связь вопроса и тега): Позволяет присваивать вопросу несколько тегов.

5 MediaAsset (Медиафайл): Хранит метаданные медиафайлов (изображений, аудио), привязанных к вопросу (URL, тип, размер, контрольная сумма).

6 Hint (Подсказка): Подсказки, связанные с вопросом, с указанием времени их открытия во время раунда.

7 FuzzyAlias (Алиас/Синоним): Альтернативные варианты правильного ответа, заданные автором вопроса для нечеткого сопоставления.

Подсистема модерации контента:

1 ModerationQueue (Очередь модерации): Фиксирует факт подачи вопроса на модерацию, связывая вопрос, пользователя-автора и назначая статус (PENDING, APPROVED, REJECTED).

2 ModerationAction (Действие модератора): Журналирует все действия модераторов (утверждение, отклонение, редактирование) с указанием причины и сохранением снимка изменений.

Игровая подсистема:

1 Room (Комната): Виртуальное пространство для игры. Хранит настройки: название, приватность, лимит игроков, условие победы, режим выбора тегов.

2 RoomTagSelection (Выбор тегов комнаты): Определяет, какие теги и с каким весом используются для подбора вопросов в конкретной комнате.

3 Invitation (Приглашение): Хранит коды и ссылки для доступа в приватные комнаты.

4 GameSession (Игровая сессия): Сущность, представляющая собой одну полную игру внутри комнаты, от старта до завершения.

5 PlayerInRoom (Игрок в комнате): Регистрирует участие пользователя в конкретной игровой сессии.

6 GameRound (Раунд игры): В рамках сессии хранит данные каждого раунда: выпавший вопрос, временные рамки.

7 PlayerAnswer (Ответ игрока): Фиксирует каждую попытку ответа пользователя в раунде с временной меткой.

Подсистема проверки ответов и аналитики:

1 AnswerEvaluation (Оценка ответа): Ядро системы нечеткого сопоставления. Хранит результат проверки каждого ответа: был ли он засчитан, какая стратегия использовалась (EXACT, LEVENSHTEIN, SEMANTIC), уверенность алгоритма и начисленные очки.

2 Scoreboard (Таблица очков): Аккумулирует общие результаты пользователя в рамках одной игровой сессии.

3 UserStats (Статистика пользователя): Агрегированные персональные метрики игрока за все время (сыграно игр, точность, среднее время ответа).

4 LeaderboardEntry (Запись в рейтинге): Рассчитываемые позиции пользователей в глобальных, недельных и сезонных рейтингах.

Связи между ключевыми сущностями:

1 Пользователь (User) может создавать вопросы (Question) и участвовать в игровых сессиях (GameSession) как игрок (PlayerInRoom).

2 Вопрос (Question) проходит жизненный цикл через очередь модерации (ModerationQueue), где модераторы (User) совершают действия (ModerationAction).

3 Игровая сессия (GameSession) состоит из раундов (GameRound), в каждом из которых игроки (PlayerInRoom) дают ответы (PlayerAnswer), которые оцениваются (AnswerEvaluation) и влияют на общий счет (Scoreboard).

Данная инфологическая модель адекватно отражает все бизнес-процессы приложения QuizFuzz и служит основой для перехода к даталогическому проектированию.

* 1. Даталогическая модель

Даталогическая модель описывает структуру данных на уровне базы данных, в том числе типы данных, ограничения и связи между таблицами. Она отвечает за определение точных типов данных для каждого атрибута сущности, установление ограничений на значения и обеспечение целостности данных. В данном разделе представлена детальная спецификация всех сущностей, их атрибутов и типов данных, которые будут использоваться в базе данных системы. Модель строится на основе инфологической модели, уточняя типы данных и устанавливая точные форматы хранения информации для эффективного функционирования системы.

Описание таблиц и их атрибутов:

1 User (Пользователь):

– id (UUID, PRIMARY KEY) – уникальный идентификатор пользователя;

– email (VARCHAR(255), UNIQUE) – электронная почта;

– username (VARCHAR(100), UNIQUE) – имя пользователя;

– password\_hash (TEXT) – хэш пароля;

– created\_at (TIMESTAMPTZ) – дата регистрации;

– last\_login\_at (TIMESTAMPTZ) – дата последнего входа;

– is\_banned (BOOLEAN) – статус блокировки.

2 Role (Роль):

– id (SMALLSERIAL, PRIMARY KEY) – идентификатор роли;

– name (VARCHAR(50), UNIQUE) – название роли (Admin, Moderator, User).

3 UserRole (Связь пользователя и роли):

– user\_id (UUID, FOREIGN KEY – User) – идентификатор пользователя;

– role\_id (SMALLINT, FOREIGN KEY – Role) – идентификатор роли;

PRIMARY KEY (user\_id, role\_id).

4 QuestionType (Тип вопроса):

– id (SMALLSERIAL, PRIMARY KEY) – идентификатор типа;

– code (VARCHAR(20), UNIQUE) – код типа (TEXT, IMAGE, AUDIO);

– description (TEXT) – описание типа.

5 Question (Вопрос):

– id (UUID, PRIMARY KEY) – уникальный идентификатор вопроса;

– type\_id (SMALLINT, FOREIGN KEY – QuestionType) – тип вопроса;

– title (TEXT) – заголовок вопроса;

– prompt\_text (TEXT) – текст вопроса;

– correct\_answer (TEXT) – правильный ответ;

– difficulty (SMALLINT) – сложность (1-5);

– status (VARCHAR(20)) – статус модерации;

– author\_user\_id (UUID, FOREIGN KEY – User) – автор вопроса;

– created\_at (TIMESTAMPTZ) – дата создания;

– updated\_at (TIMESTAMPTZ) – дата обновления.

6 Tag (Тег):

– id (SERIAL, PRIMARY KEY) – идентификатор тега;

– name (VARCHAR(50), UNIQUE) – название тега;

– description (TEXT) – описание тега;

– is\_active (BOOLEAN) – активен ли тег.

7 QuestionTag (Связь вопроса и тега):

– question\_id (UUID, FOREIGN KEY – Question) – идентификатор вопроса;

– tag\_id (INTEGER, FOREIGN KEY – Tag) – идентификатор тега;

PRIMARY KEY (question\_id, tag\_id).

8 MediaAsset (Медиафайл):

– id (UUID, PRIMARY KEY) – идентификатор медиафайла;

– question\_id (UUID, FOREIGN KEY– Question) – связанный вопрос;

– media\_type (VARCHAR(20)) – тип медиа (IMAGE, AUDIO, VIDEO);

– url (TEXT) – ссылка на файл;

– storage\_provider (VARCHAR(50)) – провайдер хранилища;

– checksum (TEXT) – контрольная сумма файла;

– duration\_sec (INTEGER) – длительность (для аудио/видео);

– width (INTEGER) – ширина (для изображений);

– height (INTEGER) – высота (для изображений);

– is\_sensitive (BOOLEAN) – пометка "чувствительный контент".

9 Hint (Подсказка):

– id (UUID, PRIMARY KEY) – идентификатор подсказки;

– question\_id (UUID, FOREIGN KEY – Question) – связанный вопрос;

– order\_index (SMALLINT) – порядковый номер;

– hint\_text (TEXT) – текст подсказки;

– media\_asset\_id (UUID, FOREIGN KEY – MediaAsset) – медиа-подсказка;

– reveal\_time\_sec (SMALLINT) – время открытия в секундах.

10 FuzzyAlias (Алиас ответа):

– id (UUID, PRIMARY KEY) – идентификатор алиаса;

– question\_id (UUID, FOREIGN KEY – Question) – связанный вопрос;

– alias\_text (TEXT) – текст алиаса;

– kind (VARCHAR(20)) – тип (SYNONYM, ALT\_SPELLING, TRANSLIT).

11 ModerationQueue (Очередь модерации):

– id (UUID, PRIMARY KEY) – идентификатор записи;

– question\_id (UUID, FOREIGN KEY – Question) – проверяемый вопрос;

– submitted\_by\_user\_id (UUID, FOREIGN KEY – User) – автор вопроса;

– status (VARCHAR(20)) – статус (PENDING, APPROVED, REJECTED);

– moderator\_user\_id (UUID, FOREIGN KEY – User) – модератор;

– comment (TEXT) – комментарий модератора;

– created\_at (TIMESTAMPTZ) – дата подачи;

– updated\_at (TIMESTAMPTZ) – дата решения.

12 ModerationAction (Действие модератора):

– id (UUID, PRIMARY KEY) – идентификатор действия;

– queue\_id (UUID, FOREIGN KEY – ModerationQueue) – запись в очереди;

– action (VARCHAR(20)) – действие (APPROVE, REJECT, EDIT);

– moderator\_user\_id (UUID, FOREIGN KEY – User) – модератор;

– reason (TEXT) – причина действия;

– created\_at (TIMESTAMPTZ) – дата действия;

– diff\_snapshot (JSONB) – снимок изменений.

13 Room (Комната):

– id (UUID, PRIMARY KEY) – идентификатор комнаты;

– owner\_user\_id (UUID, FOREIGN KEY – User) – создатель;

– name (VARCHAR(255)) – название комнаты;

– is\_private (BOOLEAN) – приватная/публичная;

– access\_code\_hash (TEXT) – хэш кода доступа;

– max\_players (SMALLINT) – макс. игроков;

– victory\_condition\_type (VARCHAR(20)) – условие победы;

– victory\_value (INTEGER) – значение условия;

– selected\_tags\_mode (VARCHAR(20)) – режим тегов;

– state (VARCHAR(20)) – состояние комнаты;

– created\_at (TIMESTAMPTZ) – дата создания;

– updated\_at (TIMESTAMPTZ) – дата обновления.

14 RoomTagSelection (Выбор тегов комнаты):

– room\_id (UUID, FOREIGN KEY – Room) – комната;

– tag\_id (INTEGER, FOREIGN KEY – Tag) – тег;

– weight (SMALLINT) – вес тега;

PRIMARY KEY (room\_id, tag\_id).

15 Invitation (Приглашение):

– id (UUID, PRIMARY KEY) – идентификатор приглашения;

– room\_id (UUID, FOREIGN KEY – Room) – комната;

– code (VARCHAR(50), UNIQUE) – код приглашения;

– created\_at (TIMESTAMPTZ) – дата создания;

– expires\_at (TIMESTAMPTZ) – срок действия;

– created\_by\_user\_id (UUID, FOREIGN KEY – User) – создатель.

16 GameSession (Игровая сессия):

– id (UUID, PRIMARY KEY) – идентификатор сессии;

– room\_id (UUID, FOREIGN KEY – Room) – комната;

– started\_at (TIMESTAMPTZ) – время начала;

– ended\_at (TIMESTAMPTZ) – время окончания;

– total\_rounds\_planned (SMALLINT) – планируемое число раундов;

– total\_rounds\_played (SMALLINT) – сыгранное число раундов.

17 PlayerInRoom (Игрок в комнате):

– id (UUID, PRIMARY KEY) – идентификатор участия;

– session\_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;

– user\_id (UUID, FOREIGN KEY – User) – игрок;

– joined\_at (TIMESTAMPTZ) – время присоединения;

– left\_at (TIMESTAMPTZ) – время выхода;

– is\_owner\_snapshot (BOOLEAN) – был ли создателем.

18 GameRound (Раунд игры):

– id (UUID, PRIMARY KEY) – идентификатор раунда;

– session\_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;

– round\_index (SMALLINT) – номер раунда;

– question\_id (UUID, FOREIGN KEY – Question) – вопрос раунда;

– started\_at (TIMESTAMPTZ) – время начала;

– ended\_at (TIMESTAMPTZ) – время окончания;

– time\_limit\_sec (SMALLINT) – лимит времени.

19 PlayerAnswer (Ответ игрока):

– id (UUID, PRIMARY KEY) – идентификатор ответа;

– round\_id (UUID, FOREIGN KEY – GameRound) – раунд;

– user\_id (UUID, FOREIGN KEY – User) – игрок;

– answer\_text (TEXT) – текст ответа;

– answer\_time\_ms (INTEGER) – время ответа;

– created\_at (TIMESTAMPTZ) – время отправки.

20 AnswerEvaluation (Оценка ответа):

– id (UUID, PRIMARY KEY) – идентификатор оценки;

– player\_answer\_id (UUID, FOREIGN KEY – PlayerAnswer) – оцененный ответ;

– is\_correct (BOOLEAN) – правильность;

– match\_strategy (VARCHAR(20)) – стратегия совпадения;

– score\_awarded (INTEGER) – начисленные очки;

– confidence (NUMERIC(3,2)) – уверенность алгоритма;

– normalized\_answer (TEXT) – нормализованный ответ;

– evaluated\_at (TIMESTAMPTZ) – время оценки.

21 Scoreboard (Таблица очков):

– id (UUID, PRIMARY KEY) – идентификатор записи;

– session\_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;

– user\_id (UUID, FOREIGN KEY – User) – игрок;

– score\_total (INTEGER) – общий счет;

– correct\_count (INTEGER) – число верных ответов;

– unique\_correct\_count (INTEGER) – уникальные верные ответы.

22 UserStats (Статистика пользователя):

– user\_id (UUID, PRIMARY KEY, FOREIGN KEY – User) – пользователь;

– games\_played (INTEGER) – сыграно игр;

– games\_won (INTEGER) – выиграно игр;

– questions\_answered (INTEGER) – дано ответов;

– unique\_first\_correct (INTEGER) – первых правильных ответов;

– avg\_answer\_time\_ms (INTEGER) – среднее время ответа;

– favorite\_tags (JSONB) – любимые теги;

– top\_accuracy\_tags (JSONB) – теги с лучшей точностью;

– updated\_at (TIMESTAMPTZ) – время обновления.

23 LeaderboardEntry (Запись в рейтинге):

– id (UUID, PRIMARY KEY) – идентификатор записи;

– period (VARCHAR(20)) – период (GLOBAL, WEEKLY, SEASON);

– start\_at (TIMESTAMPTZ) – начало периода;

– end\_at (TIMESTAMPTZ) – конец периода;

– user\_id (UUID, FOREIGN KEY – User) – пользователь;

– score (INTEGER) – очки в рейтинге;

– rank (INTEGER) – позиция в рейтинге.

24 AuditLog (Журнал аудита):

– id (UUID, PRIMARY KEY) – идентификатор записи;

– user\_id (UUID, FOREIGN KEY – User) – пользователь;

– action (VARCHAR(100)) – действие;

– entity\_type (VARCHAR(50)) – тип сущности;

– entity\_id (UUID) – идентификатор сущности;

– meta (JSONB) – метаданные;

– created\_at (TIMESTAMPTZ) – время события.

Созданная даталогическая модель обеспечивает надежную основу для физической реализации базы данных. Все сущности имеют четко определенные типы данных и ограничения целостности, что гарантирует согласованность хранимой информации. Связи между таблицами реализованы через внешние ключи, что поддерживает реляционную целостность данных на уровне СУБД.

1. РАЗРАБОТКА БАЗЫ ДАННЫХ

Разработка базы данных представляет собой ключевой этап создания программного комплекса QuizFuzz, обеспечивающий основу для хранения, обработки и анализа всей информации системы. На данном этапе осуществляется переход от даталогической модели к физической реализации средствами выбранной СУБД PostgreSQL.

База данных предназначена для централизованного хранения сведений о пользователях, вопросах, игровых сессиях, ответах участников, статистике и модерации контента. Она обеспечивает взаимосвязанность всех сущностей, целостность данных, а также возможность выполнения аналитических запросов и поддержки игрового процесса в реальном времени.

* 1. Физическая модель базы данных

Физическая модель базы данных представляет собой реализацию даталогической модели в конкретной СУБД PostgreSQL. Модель включает таблицы, индексы, последовательности и ограничения целостности, оптимизированные для эффективного выполнения запросов и обеспечения надежности хранения данных.

Ключевые аспекты физической модели:

1 Типы данных – Использование соответствующих типов данных PostgreSQL: UUID для первичных ключей, TIMESTAMPTZ для временных меток, JSONB для гибкого хранения структурированных данных, SMALLINT для оптимизации хранения числовых значений с малым диапазоном.

2 Ограничения целостности – Реализация внешних ключей с каскадными операциями, проверочные ограничения (CHECK), уникальные ограничения (UNIQUE) и ограничения NOT NULL для обеспечения логической целостности данных.

3 Индексация – Создание оптимальных индексов для ускорения выполнения запросов, включая B-tree индексы для часто используемых полей фильтрации и GIN индексы для полнотекстового поиска и работы с JSONB данными.

* 1. Создание таблиц

Создание таблиц выполнялось с помощью SQL-скриптов, реализующих физическую модель данных. Каждая таблица содержит строго типизированные поля с соответствующими ограничениями.

Пример создания основных таблиц:

-- Таблица пользователей

CREATE TABLE "User" (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

username VARCHAR(100) UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

created\_at TIMESTAMPTZ DEFAULT NOW(),

last\_login\_at TIMESTAMPTZ,

is\_banned BOOLEAN DEFAULT FALSE

);

-- Таблица вопросов

CREATE TABLE Question (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

type\_id SMALLINT NOT NULL REFERENCES QuestionType(id),

title TEXT NOT NULL,

prompt\_text TEXT,

correct\_answer TEXT NOT NULL,

difficulty SMALLINT CHECK (difficulty BETWEEN 1 AND 5),

status VARCHAR(20) DEFAULT 'DRAFT',

author\_user\_id UUID REFERENCES "User"(id),

created\_at TIMESTAMPTZ DEFAULT NOW(),

updated\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Таблица игровых сессий

CREATE TABLE GameSession (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

room\_id UUID NOT NULL REFERENCES Room(id),

started\_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

ended\_at TIMESTAMPTZ,

total\_rounds\_planned SMALLINT NOT NULL,

total\_rounds\_played SMALLINT DEFAULT 0

);

Пример создания таблиц связей:

-- Связь вопросов и тегов

CREATE TABLE QuestionTag (

question\_id UUID NOT NULL REFERENCES Question(id) ON DELETE CASCADE,

tag\_id INTEGER NOT NULL REFERENCES Tag(id) ON DELETE CASCADE,

PRIMARY KEY (question\_id, tag\_id)

);

-- Связь комнат и тегов с весами

CREATE TABLE RoomTagSelection (

room\_id UUID NOT NULL REFERENCES Room(id) ON DELETE CASCADE,

tag\_id INTEGER NOT NULL REFERENCES Tag(id) ON DELETE CASCADE,

weight SMALLINT NOT NULL DEFAULT 1,

PRIMARY KEY (room\_id, tag\_id)

);

Пример создания таблиц для системы ответов и оценки:

-- Таблица ответов игроков

CREATE TABLE PlayerAnswer (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

round\_id UUID NOT NULL REFERENCES GameRound(id) ON DELETE CASCADE,

user\_id UUID NOT NULL REFERENCES "User"(id),

answer\_text TEXT NOT NULL,

answer\_time\_ms INTEGER NOT NULL,

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Таблица оценки ответов (ядро нечеткого сопоставления)

CREATE TABLE AnswerEvaluation (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

player\_answer\_id UUID NOT NULL REFERENCES PlayerAnswer(id) ON DELETE CASCADE,

is\_correct BOOLEAN NOT NULL,

match\_strategy VARCHAR(20) NOT NULL,

score\_awarded INTEGER NOT NULL,

confidence NUMERIC(3,2) CHECK (confidence >= 0 AND confidence <= 1),

normalized\_answer TEXT NOT NULL,

evaluated\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Таблица статистики пользователей

CREATE TABLE UserStats (

user\_id UUID PRIMARY KEY REFERENCES "User"(id),

games\_played INTEGER DEFAULT 0,

games\_won INTEGER DEFAULT 0,

questions\_answered INTEGER DEFAULT 0,

unique\_first\_correct INTEGER DEFAULT 0,

avg\_answer\_time\_ms INTEGER DEFAULT 0,

favorite\_tags JSONB,

top\_accuracy\_tags JSONB,

updated\_at TIMESTAMPTZ DEFAULT NOW()

);

* 1. Запросы

Запросы к базе данных обеспечивают реализацию бизнес-логики приложения и позволяют получать необходимую информацию для отображения в пользовательском интерфейсе и проведения игрового процесса.

Примеры ключевых запросов, используемых в системе:

1 Получение списка активных публичных комнат:

SELECT r.id, r.name, r.max\_players,

COUNT(pir.user\_id) as current\_players,

u.username as owner\_name

FROM Room r

JOIN "User" u ON r.owner\_user\_id = u.id

LEFT JOIN GameSession gs ON r.id = gs.room\_id AND gs.ended\_at IS NULL

LEFT JOIN PlayerInRoom pir ON gs.id = pir.session\_id AND pir.left\_at IS NULL

WHERE r.is\_private = false

AND r.state = 'LOBBY'

GROUP BY r.id, u.username;

2 Подбор вопросов для игровой сессии по тегам комнаты:

SELECT q.id, q.title, q.prompt\_text, q.correct\_answer,

q.difficulty, qt.code as question\_type

FROM Question q

JOIN QuestionType qt ON q.type\_id = qt.id

JOIN QuestionTag qtag ON q.id = qtag.question\_id

JOIN RoomTagSelection rts ON qtag.tag\_id = rts.tag\_id

WHERE rts.room\_id = $1

AND q.status = 'APPROVED'

AND q.id NOT IN (

SELECT gr.question\_id

FROM GameRound gr

JOIN GameSession gs ON gr.session\_id = gs.id

WHERE gs.room\_id = $1

AND gs.started\_at > NOW() - INTERVAL '7 days'

)

GROUP BY q.id, qt.code

ORDER BY RANDOM()

LIMIT 20;

3 Расчет результатов раунда с учетом нечеткого сопоставления:

SELECT u.username,

SUM(ae.score\_awarded) as round\_score,

ae.match\_strategy,

pa.answer\_text,

ae.normalized\_answer

FROM PlayerAnswer pa

JOIN AnswerEvaluation ae ON pa.id = ae.player\_answer\_id

JOIN "User" u ON pa.user\_id = u.id

WHERE pa.round\_id = $1

AND ae.is\_correct = true

ORDER BY ae.score\_awarded DESC, pa.answer\_time\_ms ASC;

4 Обновление статистики пользователя после завершения игры:

WITH game\_stats AS (

SELECT

pir.user\_id,

COUNT(DISTINCT gs.id) as games\_played,

COUNT(DISTINCT CASE WHEN sb.score\_total = max\_scores.max\_score THEN gs.id END) as games\_won,

COUNT(pa.id) as questions\_answered,

AVG(pa.answer\_time\_ms) as avg\_time

FROM PlayerInRoom pir

JOIN GameSession gs ON pir.session\_id = gs.id

JOIN PlayerAnswer pa ON pir.user\_id = pa.user\_id AND pa.round\_id IN (

SELECT id FROM GameRound WHERE session\_id = gs.id

)

JOIN Scoreboard sb ON pir.session\_id = sb.session\_id AND pir.user\_id = sb.user\_id

JOIN (

SELECT session\_id, MAX(score\_total) as max\_score

FROM Scoreboard

GROUP BY session\_id

) max\_scores ON sb.session\_id = max\_scores.session\_id

WHERE gs.ended\_at IS NOT NULL

AND pir.user\_id = $1

GROUP BY pir.user\_id

)

INSERT INTO UserStats (user\_id, games\_played, games\_won, questions\_answered, avg\_answer\_time\_ms, updated\_at)

SELECT user\_id, games\_played, games\_won, questions\_answered, avg\_time, NOW()

FROM game\_stats

ON CONFLICT (user\_id)

DO UPDATE SET

games\_played = EXCLUDED.games\_played,

games\_won = EXCLUDED.games\_won,

questions\_answered = EXCLUDED.questions\_answered,

avg\_answer\_time\_ms = EXCLUDED.avg\_answer\_time\_ms,

updated\_at = NOW();

В результате проектирования и реализации базы данных на основе PostgreSQL была создана оптимальная, устойчивая и расширяемая структура, которая обеспечивает целостность, быстродействие и удобство работы с данными. Использование реляционной модели позволило эффективно описать все взаимосвязи между элементами системы и обеспечить их согласованность при выполнении любых операций.

ЗаКЛючение

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А (обязательное) Справка о проверке на заимствование

# ПРИЛОЖЕНИЕ Б (обязательное) Листинг программного кода

# ПРИЛОЖЕНИЕ В (обязательное) Функциональная схема алгоритма, реализующего программное средство

# ПРИЛОЖЕНИЕ Г (обязательное) Блок схема алгоритма, реализующего программное средство

# ПРИЛОЖЕНИЕ Д (обязательное) Графический интерфейс пользователя

# ПРИЛОЖЕНИЕ Е (обязательное) Ведомость документов